

**UNIVERSIDAD DE SAN CARLOS DE GUATEMALA**

**FACULTAD DE INGENIERIA**

**CIENCIAS Y SISTEMAS**

**ARQUITECTURA DE COMPUTADORES 1**

**AUX. JORGE GUTIERREZ**

**ING. OTTO LEIVA**



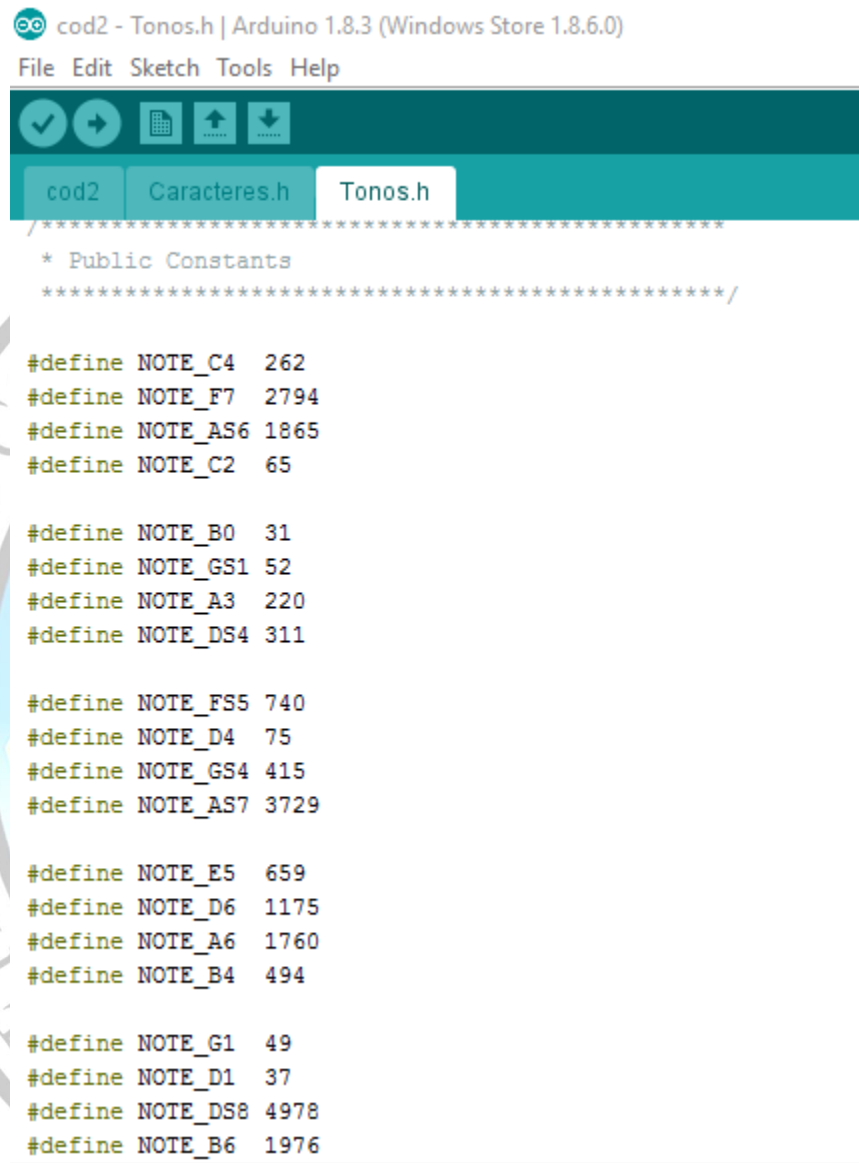
**MANUAL TECNICO PRACTICA 1 ARDUINO**

**INTEGRANTES**

<b>OSCAR RENE CUELLAR MANCILLA</b>	<b>201503712</b>
<b>HAYRTON OMAR IXPATÁ COLOCH</b>	<b>201313875</b>
<b>NERY GONZALO GALVEZ GÓMEZ</b>	<b>201403525</b>
<b>YOSELIN ANNELICE LEMUS LÓPEZ</b>	<b>201403819</b>

## EXPLICACION DE CODIGO:

En la siguiente cabecera se definieron todos los valores de las notas que se utilizan para la generación de las melodías que se reproducen al completar cada nivel del juego de la ruleta.



```
cod2 - Tonos.h | Arduino 1.8.3 (Windows Store 1.8.6.0)
File Edit Sketch Tools Help

cod2 Caracteres.h Tonos.h

/*****
 * Public Constants
 *****/

#define NOTE_C4 262
#define NOTE_F7 2794
#define NOTE_AS6 1865
#define NOTE_C2 65

#define NOTE_B0 31
#define NOTE_GS1 52
#define NOTE_A3 220
#define NOTE_DS4 311

#define NOTE_FS5 740
#define NOTE_D4 75
#define NOTE_GS4 415
#define NOTE_AS7 3729

#define NOTE_E5 659
#define NOTE_D6 1175
#define NOTE_A6 1760
#define NOTE_B4 494

#define NOTE_G1 49
#define NOTE_D1 37
#define NOTE_DS8 4978
#define NOTE_B6 1976
```

Con la cabecera creada se necesitó incluir dentro del programa principal esta cabecera para tener acceso a la información de la misma. Además de crear una constante con el número del pin (para la reproducción de ondas análogas se hizo uso de uno de los pines marcados como PWM) a utilizarse para la reproducción del sonido a través de un buzzer.

```
#include <FrequencyTimer2.h>
#include "Tonos.h"
#include "Caracteres.h"

const int alarma = 13;
```

Así mismo se configuro el pin para que fuera una salida dentro del método "Setup"

```
void setup() {  
  
    Serial.begin(9600);  
    //PINES DE ENTRADA  
    pinMode(boton, INPUT);  
    pinMode(interruptor, INPUT);  
    //PINES DE SALIDA  
    pinMode(alarma, OUTPUT);  
}
```

Llamado de la función "Buzzer" se hace dentro de la función "validar()", la cual ejecuta el Buzzer

```
void verificar(){  
    estadoboton = digitalRead(boton);  
    if((estadoboton == HIGH) && (estadoAnt == LOW)){  
        estado = 1 - estado;  
        delay(40);  
    }  
    estadoAnt = estadoboton;  
  
    if((estadoboton == HIGH) && (acierto == true)){  
        tSonido = millis();  
        Nivel++;  
        Buzzer(Nivel);  
        velocidad = velocidad + 20;  
    }  
}
```

Para la creación de las melodías se creó un método llamado "Buzzer" donde se asignan los valores que se reproducirán dentro del array "Cancion [ ]", dependiendo del número de nivel que se le mande como parámetro y posteriormente se hace un llamado a la función "Reproducir()"

```

void Buzzer(int Nivel){
  switch(Nivel){
    case 1:
      Cancion[0]= NOTE_C4; Cancion[1]= NOTE_F7; Cancion[2]= NOTE_AS6; Cancion[3]= NOTE_C2;
      Reproducir();
      break;
    case 2:
      Cancion[0]= NOTE_B0; Cancion[1]= NOTE_GS1; Cancion[2]= NOTE_A3; Cancion[3]= NOTE_DS4;
      Reproducir();
      break;
    case 3:
      Cancion[0]= NOTE_FS5; Cancion[1]= NOTE_D4; Cancion[2]= NOTE_GS4; Cancion[3]= NOTE_AS7;
      Reproducir();
      break;
    case 4:
      Cancion[0]= NOTE_E5; Cancion[1]= NOTE_D6; Cancion[2]= NOTE_A6; Cancion[3]= NOTE_B4;
      Reproducir();
      break;
    case 5:
      Cancion[0]= NOTE_G1; Cancion[1]= NOTE_D1; Cancion[2]= NOTE_DS8; Cancion[3]= NOTE_B6;
      Reproducir();
      break;
  }
}

```

Dentro del método “Reproducir()” se hace un recorrido del array “Cancion [ ] ” y se envía cada una de las notas al buzzer por medio de la función “analogWrite(#pin,Valor ).

```

void Reproducir() {
  for(int Nota=0;Nota < 4; Nota++){
    analogWrite(alarma,Cancion[Nota]);
    delay(200);
  }
  analogWrite(alarma,0);
}

```

Para el pintado de la matriz se utilizó una librería llamada FrequencyTimer2, la cual nos ofrece gracias a su método setOnOverflow(); hacer de un método un hilo, el cual se ejecuta indefinidamente cada cierto tiempo el cual es definido en microsegundos y se le asigna con el método setPeriod();

```

FrequencyTimer2::disable();
FrequencyTimer2::setPeriod(200);
FrequencyTimer2::setOnOverflow(display);

```

El método display() es el que va habilitando los pines de la matriz respecto a un arreglo de 8x8 que representa a la matriz, en la cual los 1 indican donde debe encender led y 0 donde debe estar apagado, las led de la matriz encienden con la combinación 0 en filas y 1 en columnas que es lo que se aplica en el método que las pinta.

```
void display() {
    digitalWrite(Columnas[NumeroColumna], HIGH);
    NumeroColumna++;
    if (NumeroColumna == 8) {
        NumeroColumna = 0;
    }
    for (int NumeroFila = 0; NumeroFila < 8; NumeroFila++) {
        if (leds[NumeroColumna][NumeroFila] == 1) {
            digitalWrite(Filas[NumeroFila], HIGH);
        }
        else {
            digitalWrite(Filas[NumeroFila], LOW);
        }
    }
    digitalWrite(Columnas[NumeroColumna], LOW);
}
```

Se creo un tipo de menú en donde al momento de presionar el botón se guarda una referencia del tiempo, luego se toma nuevamente el tiempo hasta que suelte el botón y se hace una resta aritmética de los tiempos para ver el tiempo que mantuvo presionado el botón y con forme a el tomar una decisión sobre que hacer.

```
if(estadoboton == HIGH)
{
    Serial.println("boton en 1");
    tiempo=millis();
    while(estadoboton == HIGH)
    {
        tiempo2=millis();
        estadoboton = digitalRead(boton);
    }
    tiempo3=tiempo2-tiempo;
    if(tiempo3>500)
    {
        Serial.println("EN MENU: BOTON PRESIONADO MAS DE DOS SEGUNDOS");
        //EL BOTON SE PULSO POR MAS DE DOS SEGUNDOS, MOSTRAR PUNTEOS MAS ALTOS
        Display_Barrido(hora+": "+String(t1)+"-"+String(p1)+" ");
        Display_Barrido(hora+": "+String(t2)+"-"+String(p2)+" ");
        Display_Barrido(hora+": "+String(t3)+"-"+String(p3)+" ");
    }
    else
    {
        Serial.println("EN MENU: BOTON PRESIONADO MENOS DE DOS SEGUNDOS");
        //SE PRESIONO MENOS DE DOS SEGUNDOS, INICIAR JUEGO
        JUEGO();
    }
}
```

Se crearon dos métodos los cuales podían mostrar mensajes en la matriz de leds tomando como parámetro un String. Uno de ellos es Display\_Barrido() el cual va moviendo los arreglos de las letras dentro del arreglo de la matriz de 8x8 para simular un desplazamiento.

```
void Display_Barrido(String mensaje)
{
    int contador = 0;
    int NoLetras = mensaje.length();
    byte WIN[16][8][8] = {cerro,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve,W,I,N,ESPACIO,GUION,DOSPUNTOS};

    int cont = 0;
    while(cont<NoLetras)
    {
        char let = mensaje[cont];
        char let2 = mensaje[NoLetras-cont];
        int no = PosicionByChar(let);
        int no2 = PosicionByChar(let2);

        for (int l = 0; l < 8; l++) {
            if(EstadoInterruptor==true)
            {
                for (int i = 0; i < 7; i++)
                {
                    for (int j = 0; j < 8; j++)
                    {
                        leds[j][i] = leds[j][i+1];
                    }
                }
            }
        }
    }
}
```

Otro de los métodos es Display\_LxL() el cual muestra también un mensaje a travez de un String dado pero letra por letra nada más.

```
void Display_LxL(String mensaje)
{
    int NoLetras = mensaje.length();
    byte WIN[16][8][8] = {cerro,uno,dos,tres,cuatro,cinco,seis,siete,ocho,nueve,W,I,N,ESPACIO,GUION,DOSPUNTOS};

    int cont = 0;
    while(cont<NoLetras)
    {
        char let = mensaje[cont];
        int no = PosicionByChar(let);
        for (int l = 0; l < 8; l++)
        {
            for (int j = 0; j < 8; j++)
            {
                leds[j][l] = WIN[no][j][l];
            }
        }
        delay(300);
        cont++;
    }
}
```

Para el ordenamiento de los punteos se creó un método de ordenamiento el cual va comparando el ultimo punteo con el siguiente y así sucesivamente y se van cambiando valores si el anterior es más grande.

```
void ORDENAR()
{
    //ORDENAMIENTO
    if(p3>p2)
    {
        int temp = p2;
        int temp2 = t2;
        p2 = p3;
        t2 = t3;
        p3 = temp;
        t3 = temp2;
    }
    if(p2>p1)
    {
        int temp = p1;
        int temp2 = t1;
        p1 = p2;
        t1 = t2;
        p2 = temp;
        t2 = temp2;
    }
}
```

Para guardar el punteo se consulta únicamente si el ultimo punteo es menor al nuevo que ingreso, de serlo entonces se cambian valores para poder entrar al top de punteos.

```
void GuardarPunteoYTiempo(int punteo, int tiempo)
{
    if(punteo>p3)
    {
        p3 = punteo;
        t3 = tiempo;
    }
    ORDENAR();
}
```



Método principal del juego, este método verifica el sentido en que gira la ruleta dando inicio al juego y finalizándolo cuando el jugador pase los 5 niveles.

```
void JUEGO() {
  gano = false;
  tiempojuego1 = millis();
  while (gano==false)
  {
    EstadoInterruptor = digitalRead(interruptor);
    if(EstadoInterruptor==1)
    {
      giroDerecha();
    }
    else
    {
      giroIzquierda();
    }
  }
  Display_LxL("WIN"+result);
  Display_LxL("WIN"+result);

  gano = false;

  FrequencyTimer2::setPeriod(200);
}
```

Método para la que la ruleta gire hacia la derecha llamando a los métodos que imprimen la figura en la matriz de Led's, como también el método que verifica el estado del botón.

```
void giroDerecha() { //secuencia hacia la derecha
  //tiempojuego1 = millis();
  s0();          // llama el primer estado parte superior y posicion de acierto
  ruleta();
  verificar(); if (gano==true) {Serial.println("GANASTE");Serial.println("Tiempo: ");Serial.print(tiempojuego3);return 0;}
  s1();
  ruleta();
  verificar();
  s2();
  ruleta();
  verificar();
  ...
}
```

Método para la que la ruleta gire hacia la izquierda llamando a los métodos que imprimen la figura en la matriz de Led's, como también el método que verifica el estado del botón.

```
void giroIzquierda() { //secuencia hacia la izquierda
  //tiempojuego1 = millis();
  s0();          //estado inicial parte superior y posicion de acierto
  ruleta();
  verificar(); if (gano==true) {Serial.println("GANASTE");Serial.println("Tiempo: ");Serial.print(tiempojuego3);return 0;}
  s13();
  ruleta();
  verificar();
  s12();
  ruleta();
  verificar();
  ...
}
```

Método que verifica si el botón es presionado, al presionarse el botón verifica si el movimiento de la figura se encuentra en la posición de acierto para sumar puntos y subir de nivel aumentando la velocidad de la figura, de lo contrario se restaran puntos.



```

void verificar(){
  pulsador = digitalRead(push);
  if((pulsador == HIGH) && (estadoAnt == LOW)){
    estado = 1 - estado;
    delay(40);
  }
  estadoAnt = pulsador;

  if((estado == 1) && (acierto == true)){
    velocidad = velocidad - 100;
    if(velocidad == 0){
      velocidad = 600;
      tope = tope+1;
      topJugadores(puntos);
      tiempojuego2 = millis();
      tiempojuego3 = tiempojuego2 - tiempojuego1;
      Serial.print("tiempojuego: ");
      Serial.println(tiempojuego3);
      result = String(puntos) + "-" + String(tiempojuego3);
      Serial.println(result);
      puntos = 0;
      Serial.println("felicidades ganaste!!!");
    }
  }
}

```

Método ruleta este método se encarga de cambiar la velocidad cuando el jugador acierta en la posición predefinida y sumando los puntos correspondientes.

```

void ruleta() {
  FrequencyTimer2::setPeriod(velocidad);
}

```

Métodos S estos se encargan recorrer la matriz y capturar el contenido para poderlo mostrar en la matriz de led, este método es utilizado en giroDerecha(); y giroIzquierda();

```

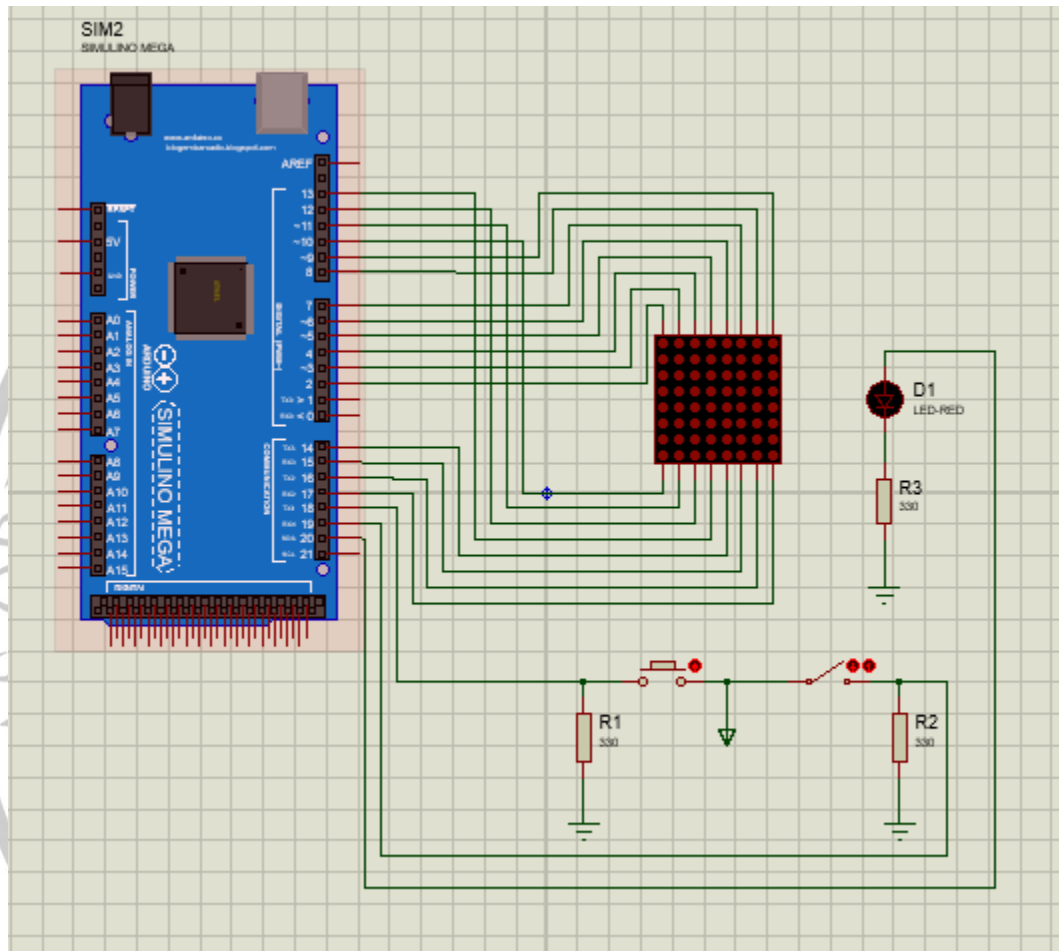
void s0() {
  acierto = true;
  for(int i=0;i<8;i++){
    for(int j=0;j<8;j++){
      matriz[i][j]=f1[i][j];
    }
  }
}

void s1() {
  acierto = false;
  for(int i=0;i<8;i++){
    for(int j=0;j<8;j++){
      matriz[i][j]=f2[i][j];
    }
  }
}

```

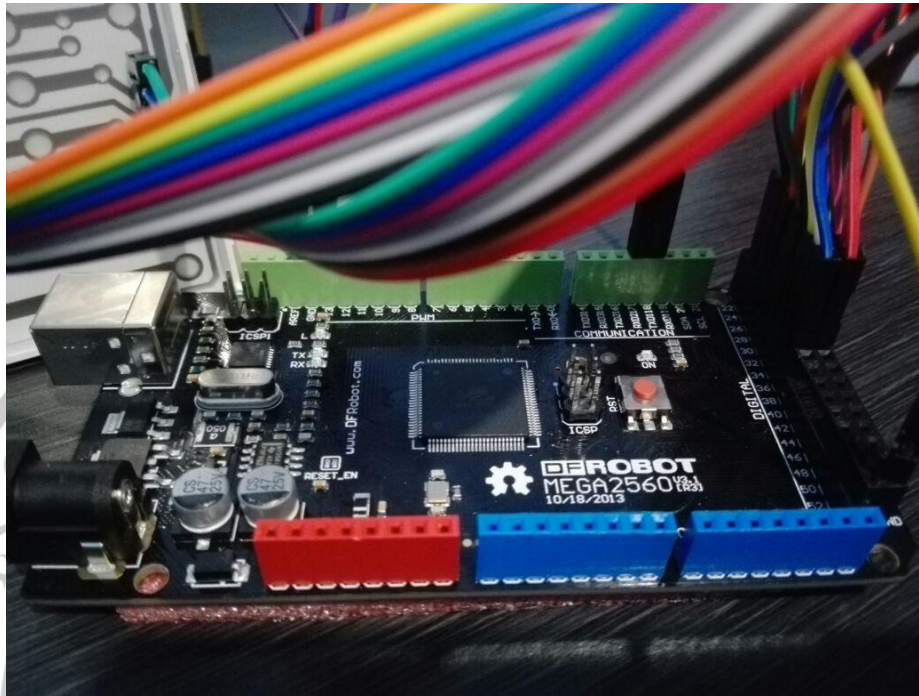
## DIAGRAMA FINAL DE LA SOLUCION:

En el siguiente diagrama se le da referencia a como está conectado el circuito dentro del encapsulado, en el cual la LED representa la alarma o Buzzer y el Switch representa al interruptor, también se muestran cómo van conectados los pines de salida del Arduino a la matriz de leds.

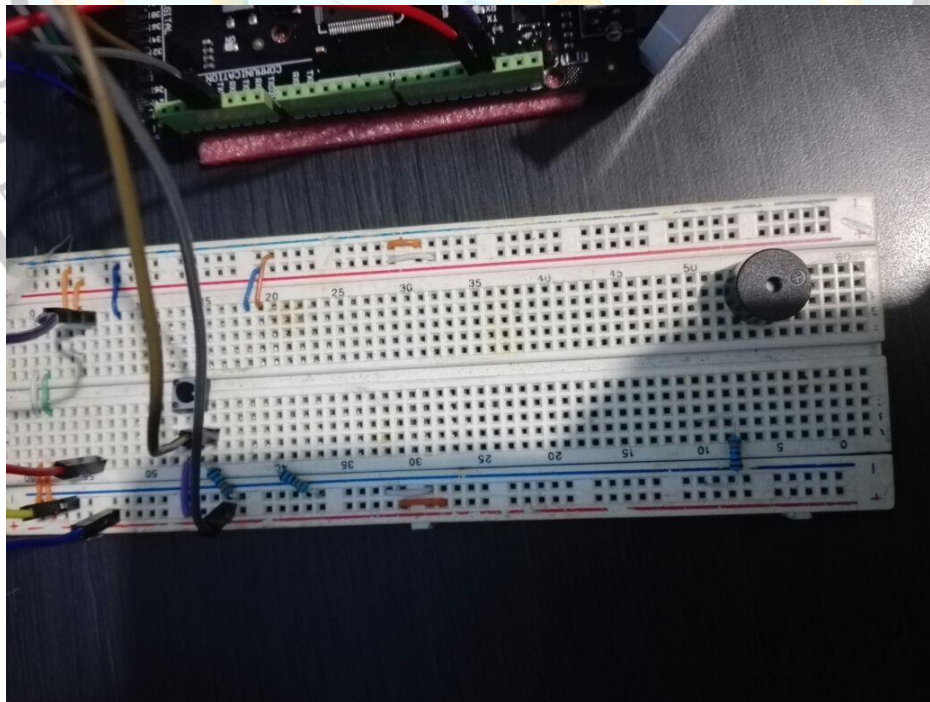


## COMPONENTES UTILIZADOS:

### ARDUINO MEGA:



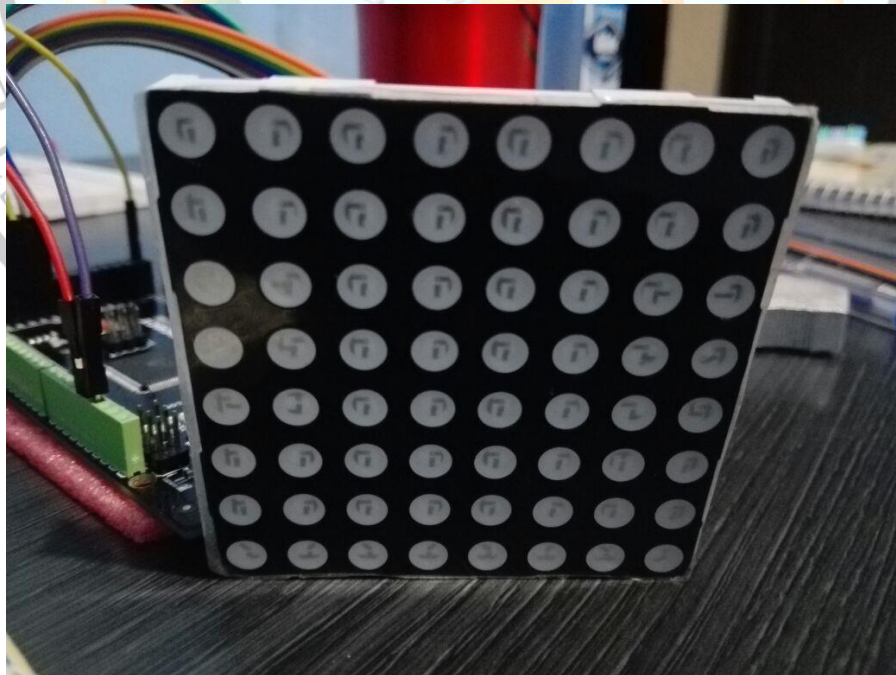
### PROTOBOARD:



**BUZZER, BOTON E INTERRUPTOR:**



**MATRIZ DE LEDS DE 8X8:**





**JUMPERS MACHO-MACHO Y HEMBRA-MACHO:**

